B. CLAIM AMENDMENTS

1. (Currently Amended) A method for distributing cryptographic requests to a plurality of cryptographic devices comprising:

receiving a cryptographic request;

determining ~~the~~ a lowest T(N); and

sending the cryptographic request to a cryptographic device with the lowest T(N);

wherein a T(N) represents an estimated time for a cryptographic device N to completely process all of a plurality of requests currently in a Q(N); and

wherein the Q(N) represents a number of requests in a queue for a cryptographic device N's request queue in a Q(N) device queue table.

2. (Currently Amended) The method of claim 1 further comprising:

determining whether there is a second cryptographic request; and

responsive to a determination that there is a second cryptographic request, determining a new lowest T(N); and

sending the second cryptographic request to a cryptographic device with the new lowest T(N).

3. (Currently Amended) The method of claim 1 further comprising:

setting an N equal to 1;

setting the T(N) equal to 0;

setting the Q(N) equal to 0;

determining whether there is another device to query; and

responsive to a determination that there is another device to query, setting the N equal to the N plus 1 and returning to the step of setting the T(N) equal to 0.

4. (Currently Amended) The method of claim 1 further comprising:

setting a current time ~~CT~~ equal to a current system time ~~CST~~; and

updating all of a plurality of estimated ~~QI~~ queue item completion times.

5. (Currently Amended) The method of claim 1 further comprising:

setting an estimated time ~~ET~~ from an ~~ET~~ estimated time table;

determining whether a ~~QI~~ queue item is ~~the~~ an only ~~QI~~ queue item in a queue; and

responsive to a determination that the ~~QI~~queue item is the only ~~QI~~ queue item in the queue, setting ~~the~~ a QI queue item timestamp to a current time ~~CT~~.

6. (Currently Amended) The method of claim 1 further comprising:

setting the N equal to 1;

determining whether the Q(N) is empty;

responsive to a determination that the Q(N) is empty, determining whether there is another cryptographic device;

responsive to a determination that there is another cryptographic device, setting the N equal to the N plus 1 and returning to the step of determining whether the Q(N) is empty;

computing a t where the t is the time a request in a ~~QI~~ queue item at the top of a queue has been processing, by subtracting ~~the~~ a time stamp from a ~~CT~~ current time;

subtracting the t from ~~the~~ a ~~QI's~~ queue item's estimated completion time;

determining whether ~~the~~ a new estimated completion time is less than or equal to zero;

responsive to a determination that the new estimated completion time is less than or equal to zero, setting ~~the~~ an estimated time to a ~~Z~~ percent of ~~the~~ an original estimated time.

7. (Currently Amended) The method of claim 1 further comprising:

identifying a cryptographic device associated with a ~~QI~~ queue item with a completed request;

determining whether there are more ~~QI's~~ queue item's in a queue for the cryptographic device N; and

responsive to a determination that there are more ~~QI's~~ queue item's in a queue for the cryptographic device N, calculating ~~the~~ a current system time and assigning the current system time to ~~the~~ a next ~~QI's~~ queue item's timestamp.

8. (Currently Amended) A programmable apparatus for balancing the load of requests for cryptographic operations sent to a plurality of identical cryptographic devices comprising:

a computer having a processor, a memory, a plurality of PCI buses, and a plurality of cryptographic devices connected to said PCI buses;

a cryptographic API installed on said computer;

a ~~loading~~ load balancing program in said cryptographic API;

a estimated completion time subroutine in said load balancing program;

wherein, said estimated completion time subroutine directs said processor to determine a lowest T(N); and

wherein, responsive to determining ~~a~~ the lowest T(N), sending a request for a cryptographic operation to a cryptographic device with the lowest T(N);

wherein a T(N) represents an estimated time for a cryptographic device N to completely process a plurality of requests currently in a Q(N); and

wherein the Q(N) represents a number of requests in a queue for the cryptographic device N's request queue in a Q(N) device queue table.

9. (Currently Amended) The programmable apparatus of claim 8 further comprising an initialization subroutine in said load balancing program that directs said processor to set the N equal to 1, set the T(N) equal to 0, and to set the Q(N) equal to 0.

10. (Currently Amended) The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that sets ~~CT~~ a current time equal to ~~CST~~ a current system time and that updates all of a plurality of estimated ~~QI~~queue item completion times.

11. (Currently Amended) The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that sets an estimated time ~~ET~~ from an ~~ET~~ estimated time table, determines whether a ~~QI~~ queue item is thean only QI queue item in a queue, and responsive to a determination that the ~~QI~~queue item is the only ~~QI~~queue item in the queue, sets the a ~~QI~~queue item timestamp to a ~~CT~~ current time.

12. (Currently Amended) The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that computes a t where the t is a time that a request in a ~~QI~~ queue item at a top of a queue has been processing, by subtracting a time stamp from a current time ~~CT~~.

13. (Currently Amended) The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that sets the N equal to 1, determines whether the Q(N) is empty, responsive to a determination that the Q(N) is empty, determines whether there is another device, responsive to a determination that there is another device, sets the N equal to the N plus 1, computes the t, where the t is a time that a request in a ~~QI~~ queue item at a top of a queue has been processing, by subtracting a time stamp from a current time ~~CT~~, subtracts the t from ~~the~~ a ~~QI's~~ queue item's estimated completion time, determines whether ~~the~~ a new estimated completion time is less than or equal to zero, and responsive to a determination that the new estimated completion time is less than or equal to zero, sets the estimated time to a Z percent of ~~the~~ an original estimated time.

14. (Currently Amended) The programmable apparatus of claim 8 further comprising a subroutine in the load balancing program that identifies a cryptographic device associated with a QI queue item with a completed request, determines whether there are more ~~QI's~~ queue item's in a queue for the cryptographic device, and responsive to a determination that there are more ~~QI's~~ queue items in the queue for the cryptographic device, calculating ~~the~~ a current system time and assigning the current system time to ~~the~~ a next ~~QI's~~ queue item's timestamp.

15. (Currently Amended) A computer readable memory for causing a computer to balance the load of requests for cryptographic operations sent to a plurality of cryptographic devices comprising:

~~a memory~~;

a load balancing program stored in said memory;

the memory, so configured by said load balancing program, responsive to receiving a request for a cryptographic operation, causes the computer to determine a lowest T(N), and to send the cryptographic request to a cryptographic device with the lowest T(N);

wherein a T(N) represents an estimated time for a cryptographic device N to completely process a plurality of requests currently in a Q(N); and

wherein the Q(N) represents a number of requests in a queue for the cryptographic device N's request queue in a Q(N) device queue table.

16. (Currently Amended) The computer readable memory of claim 15 wherein the load balancing program comprises an initialization subroutine in said load balancing program that causes said computer to set the N equal to 1, to set the T(N) equal to 0, and to set the Q(N) equal to 0.

17. (Currently Amended) The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine ~~in the load balancing program~~ that sets ~~CT~~ a current

time equal to ~~CST~~ a current system time and that updates all of a plurality of estimated ~~QI~~ queue item completion times.

18. (Currently Amended) The computer readable memory of claim 15 wherein the load balancing program further comprises a subroutine ~~in the load balancing program~~ that sets an estimated time ~~ET~~ from an ~~ET~~ estimated time table, determines whether a ~~QI~~ queue item is ~~the~~ an only ~~QI~~ queue item in a queue, and responsive to a determination that the ~~QI~~ queue item is the only ~~QI~~queue item in the queue, sets the ~~QI~~queue item timestamp to a current time~~CT~~.

19. (Currently Amended) The computer readable memory of claim 15 wherein the load balancing program comprises a subroutine in the load balancing program that computes a t where the t is a time a request in a ~~QI~~ queue item at a top of a queue has been processing, by subtracting a time stamp from a current time ~~CT~~.

20. (Currently Amended) The computer readable memory of claim 15 wherein the load balancing program further comprises a subroutine ~~in the load balancing program~~ that sets an N equal to 1, determines whether the Q(N) is empty, responsive to a determination that the Q(N) is empty, determines whether there is another device, responsive to a determination that there is another device, sets the N equal to the N plus 1, computes a t, where the t is a time that a request in a ~~QI~~ queue item at a top of a queue has been processing, by subtracting a time stamp from a current time ~~CT~~, subtracts the t from the ~~QI's~~ queue item's estimated completion time, determines whether ~~the~~ a new estimated completion time is less than or equal to zero, and responsive to a determination that the new estimated completion time is less than or equal to zero, sets ~~the~~ an estimated time to a ~~Z~~ percent of ~~the~~ an original estimated time.

21. (Currently Amended) The computer readable memory of claim 15 wherein the load balancing program further comprises a subroutine ~~in the load balancing program~~ that identifies a

cryptographic device associated with a ~~QI~~queue item with a completed request, determines whether there are more ~~QI's~~ queue item's in a queue for the cryptographic device, and responsive to a determination that there are more ~~QI's~~ queue item's in the queue for the cryptographic device, calculating ~~the~~ a current system time and assigning the current system time to ~~the~~a next ~~QI's~~ queue item's timestamp.

22. (Currently Amended) A computer implemented process to balance the load of requests for cryptographic operations sent to a plurality of cryptographic devices, comprising: using a computer, performing the following series of steps:

receiving a cryptographic request;

setting an N equal to 1;

setting a T(N) equal to 0;

setting a Q(N) equal to 0;

determining whether the Q(N) is empty;

responsive to a determination that the Q(N) is empty, determining whether there is another device;

responsive to a determination that there is another device, setting the N equal to the N plus 1 and returning to the step of determining whether the Q(N) is empty;

computing a t where the t is ~~the~~ a time a request at the a top of ~~the~~ a queue has been processing by subtracting ~~the~~ a time stamp from a ~~CT~~ current time;

subtracting the t from ~~the~~ a cryptographic request's estimated completion time;

determining whether ~~the~~ a new estimated completion time is less than or equal to zero;

responsive to a determination that the new estimated completion time is less than or equal to zero, setting ~~the~~ an estimated time to a Z percent of ~~the~~ an original estimated time;

responsive to a determination that the new estimated time is greater than zero, determining whether there is another device to query;

responsive to determining that there is another device to query, returning to the step of determining whether the Q(N) is empty; and

identifying the a cryptographic device associated with ~~the~~ a completed request;

determining whether there are more ~~QIs~~ queue items in a queue;

responsive to a determination that there are more ~~QIs~~ queue items in the queue, calculating ~~the~~ a current system time and assigning the current system time to ~~the~~ a next ~~QI's~~ queue item's timestamp;

setting ~~CT~~ a current time equal to ~~CST~~ a current system time;

updating all of a plurality of estimated completion times;

determining ~~the~~ a cryptographic device with ~~the~~ a lowest T(N); and

sending the cryptographic request to a device with the lowest T(N);

wherein the T(N) represents an estimated time for a cryptographic device N to completely process a plurality of requests currently in the Q(N);

wherein the Q(N) represents a number of requests in a queue for the cryptographic device N's request queue in a Q(N) device queue table; and

wherein the Z is determined by an administrator.

23. (Currently Amended) The computer implemented process of claim 22 further comprising:

determining whether there is another device to query;

responsive to a determination that there is another device to query, setting N equal to N plus 1 and returning to the step of setting the T(N) equal to 0.

determining whether there is a second cryptographic request;

responsive to a determination that there is a second cryptographic request, determining a new lowest T(N); and

sending the second cryptographic request to a cryptographic device with the lowest T(N).